Designing High Performance DSM Systems using InfiniBand Features

Ranjit Noronha
and
Dhabaleswar K. Panda
The Ohio State University

NBC

Outline

- Introduction
- Motivation
- Design and Implementation
- Results
- Conclusions
- Future Work

Introduction

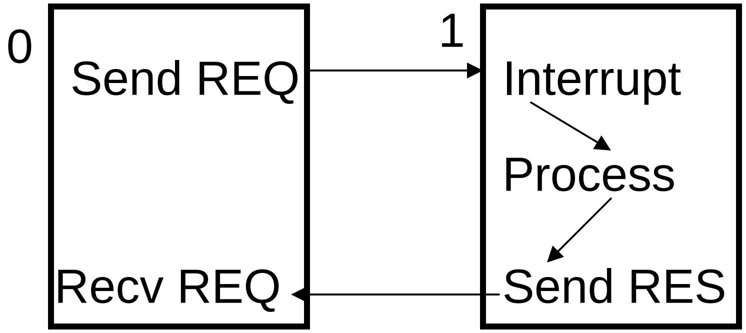
- Software DSM
  - HLRC/VIA (Rutgers), TreadMarks (Rice), JIAJIA (ICT China)
- Depends on user and software layer
- Depends on communication protocols provided by the system such as TCP, UDP, etc.
- Degraded performance because of false sharing and high overhead of communication
- Has scaling problems

Introduction

- Modern Interconnects (InfiniBand, Myrinet, Quadrics)
- Low Latency (InfiniBand 5.0 µs)
- High Bandwidth (InfiniBand 4X upto 10 Gbps)
- Programmable NIC
- User Level Protocols (VAPI, GM)
- Can deliver performance close to that of the underlying hardware
- RDMA Write/Read, Atomic Operations, Service Levels, Multicast

Motivation

- Traditional DSM
  - Uses Request / Response Communication Model (asynchronous)
  - Separate signal handler thread needed
  - Application Processing interrupted
  - Cache Effects



- Can network based features be used to reduce interrupt overhead ?

Motivation

- Asynchronous communication model
- Use network features to achieve the same effect (synchronous/hybrid communication model)
- Potential Advantages
  - Partial offload of protocol to network
  - More application processing time
  - Reduced Copying
  - Better caching
- Potential Disadvantages
  - Longer protocol execution time
  - Ordering problems
  - Consistency Issues

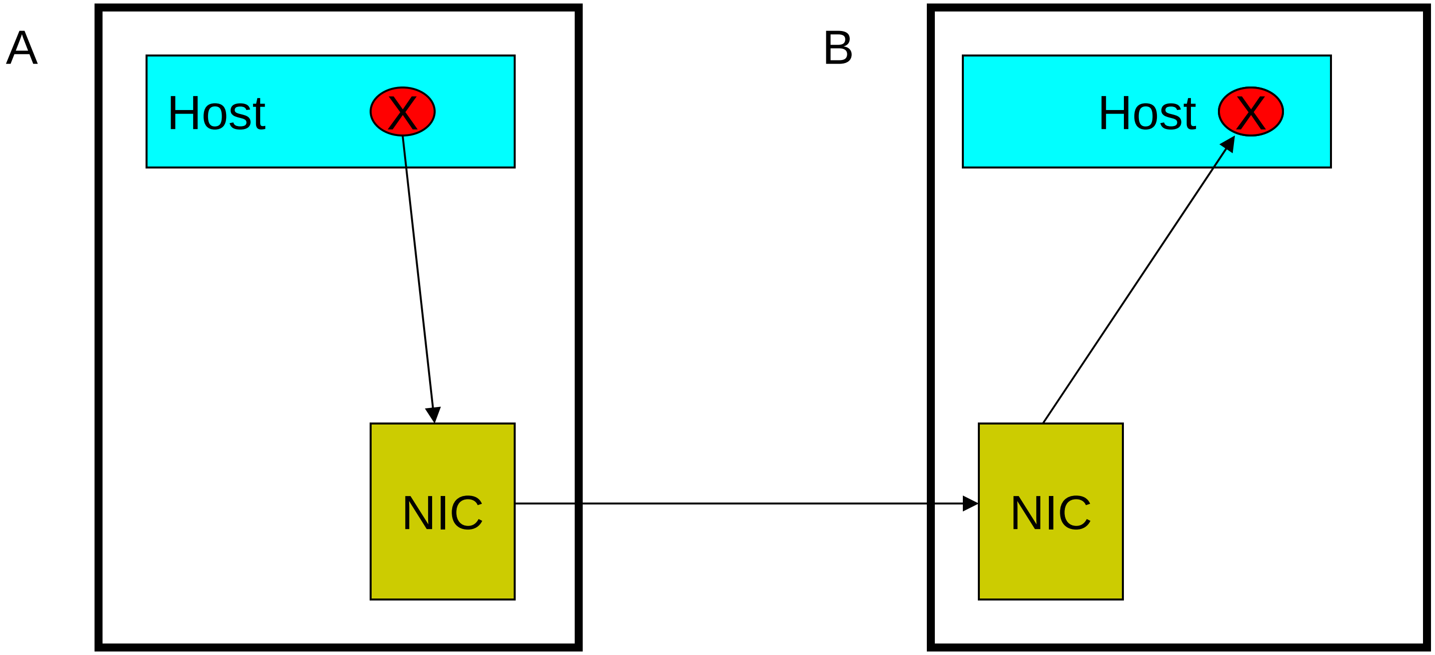
Outline

- Introduction
- Motivation
- Design and Implementation
- Results
- Conclusions
- Future Work

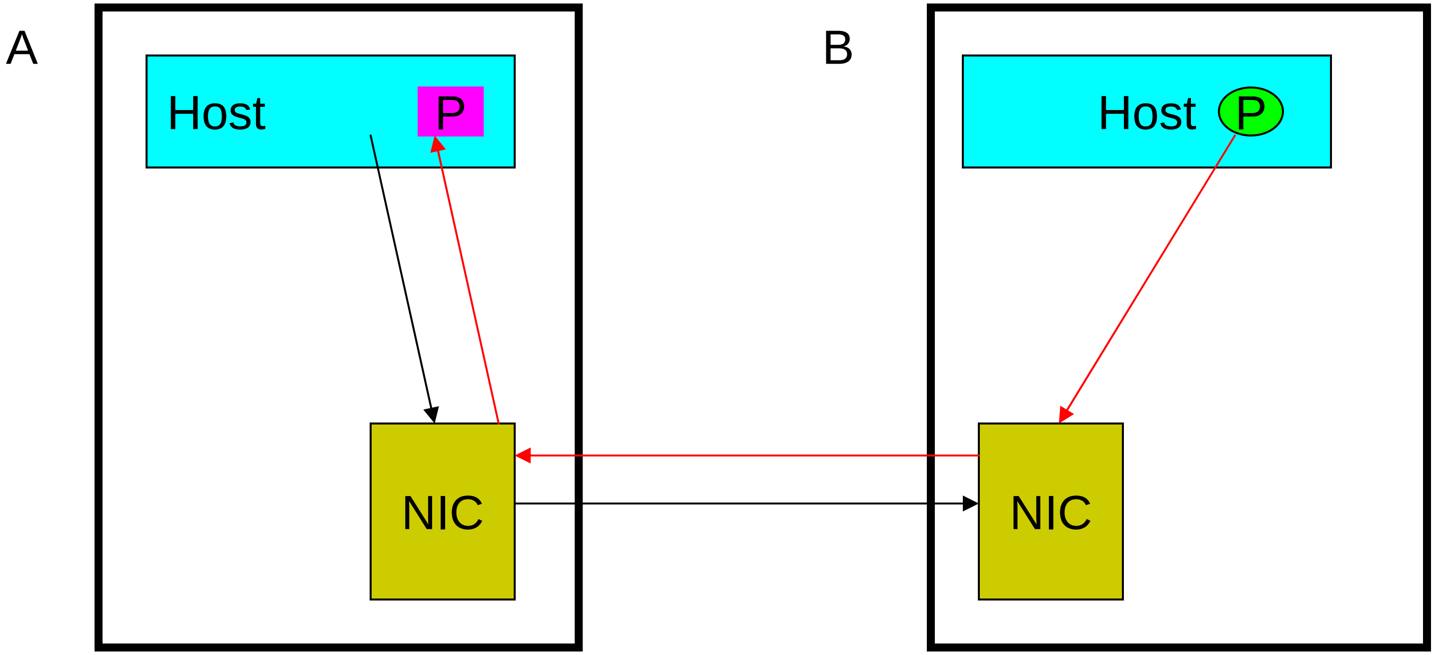
Preliminaries

- RDMA
  - Remote Direct Memory Access
  - Allows access to memory on a remote node
  - No involvement from the remote node
  - RDMA Write
  - RDMA Read

RDMA Write Example



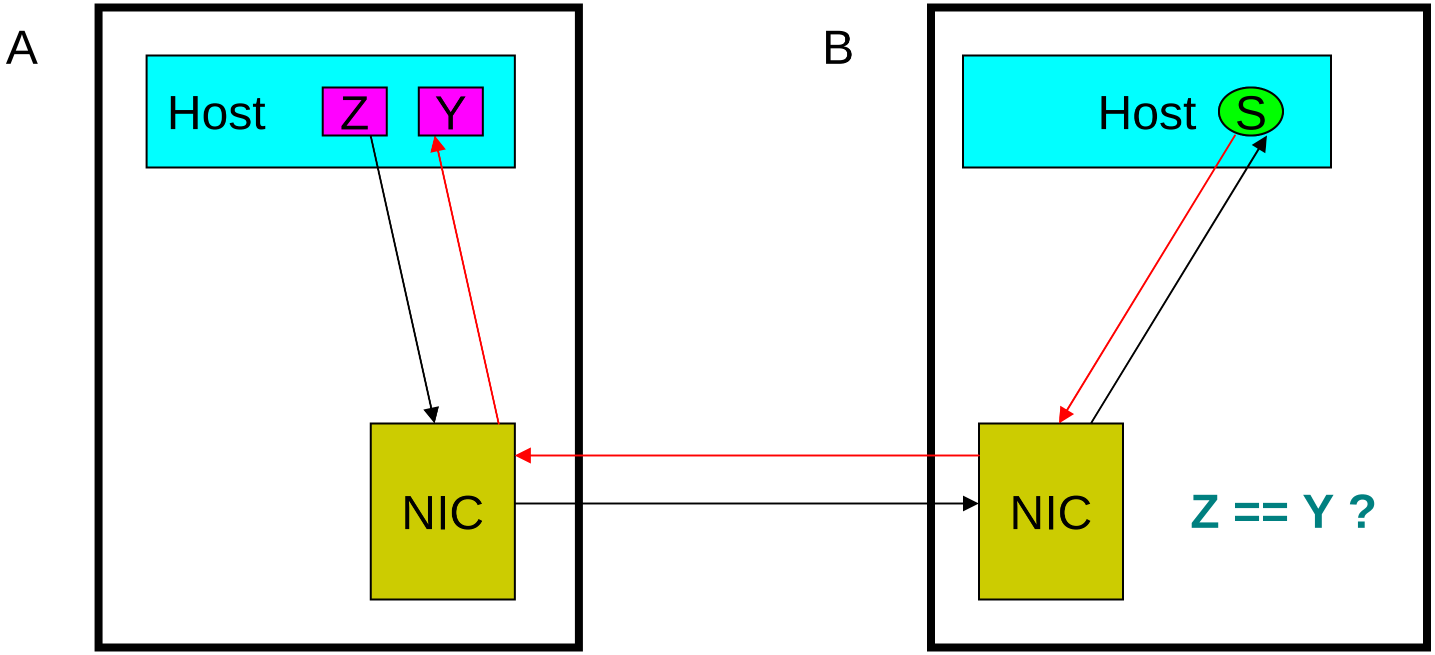
RDMA Read Example



Preliminaries - Remote Atomic Operations

- Remote Atomic Operations
  - Compare and Swap (CMP_AND_SWAP)
    - Conditionally change a location on a remote machine atomically
  - Fetch and Add

Remote Atomic Operations Example

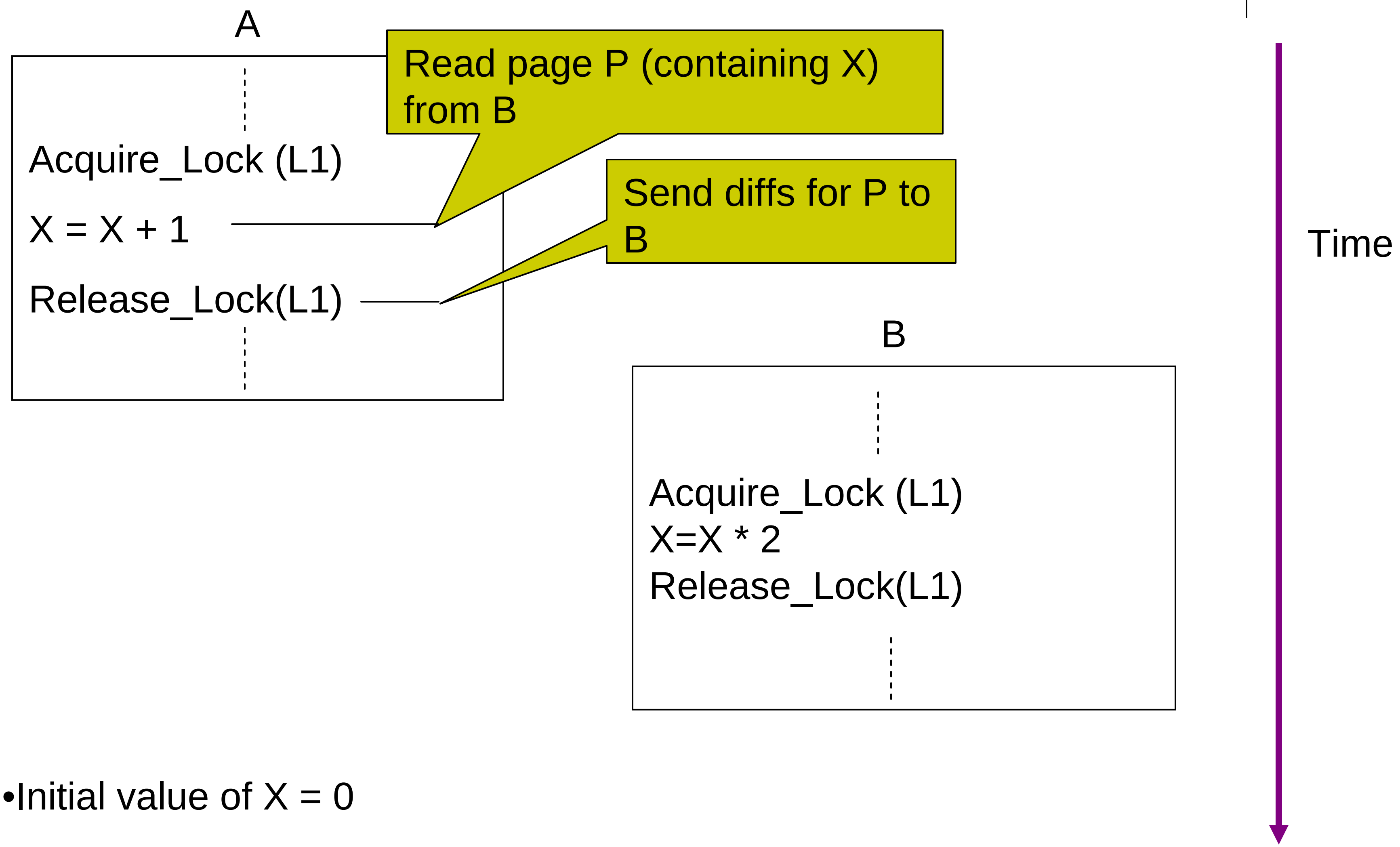


- Compare and Swap

Preliminaries - HLRC

- HLRC/VIA (Rutgers)
  - Home Based Lazy Release Consistency Model
  - Page Based DSM System
- Basic Operations
  - Page
  - Diff
  - Lock
- Use interrupts
  - Referred to as ASYNC

HLRC Programming Example



```
Acquire_Lock (L1)
X = X + 1
Release_Lock(L1)
```

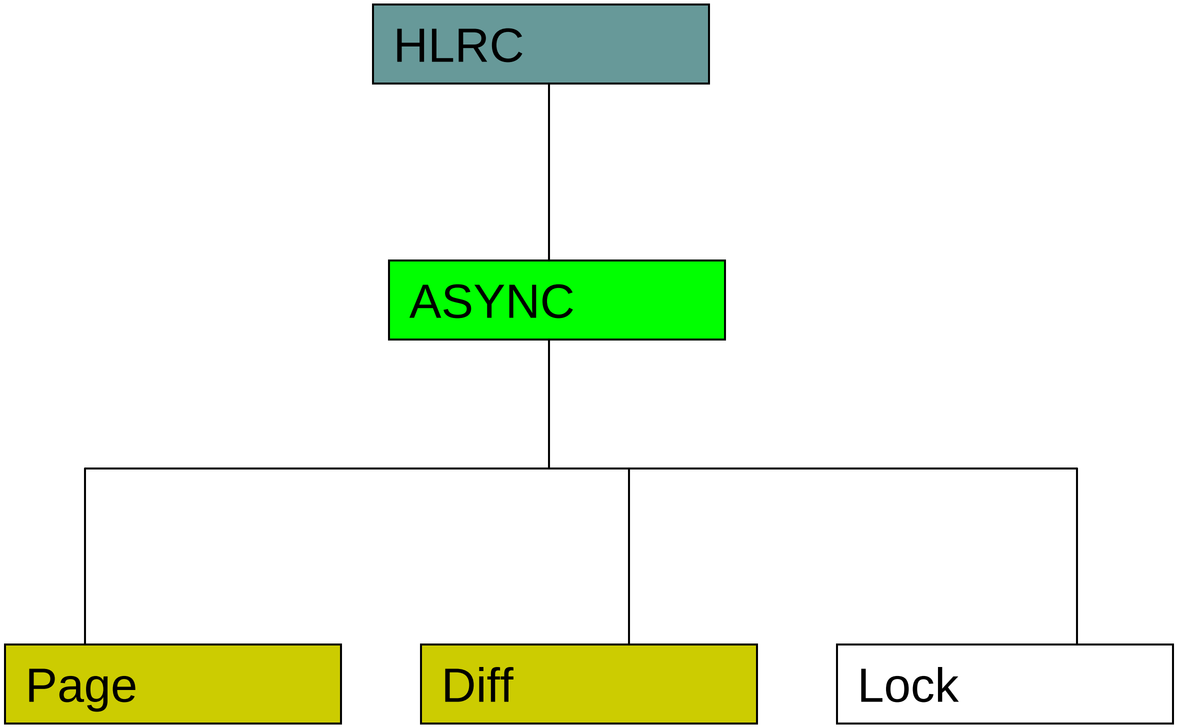
Read page P (containing X) from B

Send diffs for P to B

```
Acquire_Lock (L1)
X=X * 2
Release_Lock(L1)
```

- Initial value of X = 0
- B is home node for page P containing X

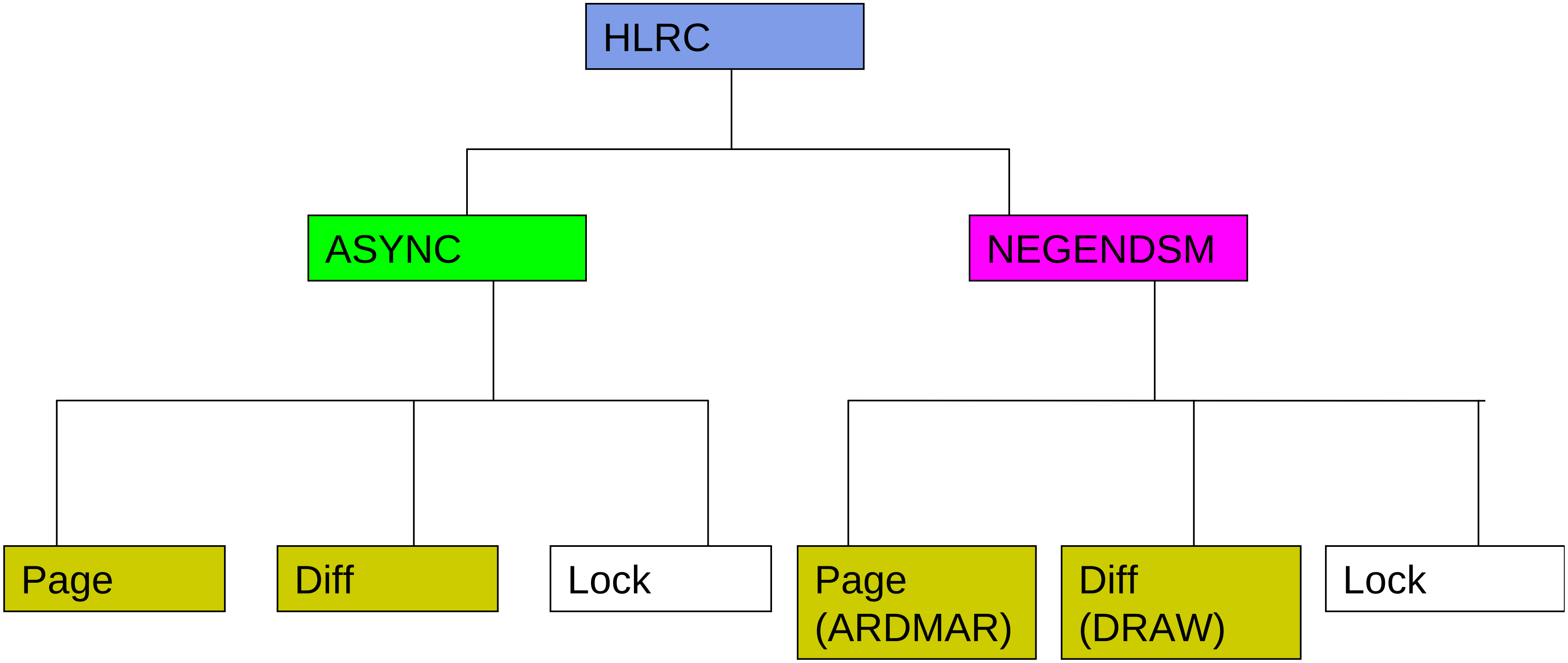
HLRC Design



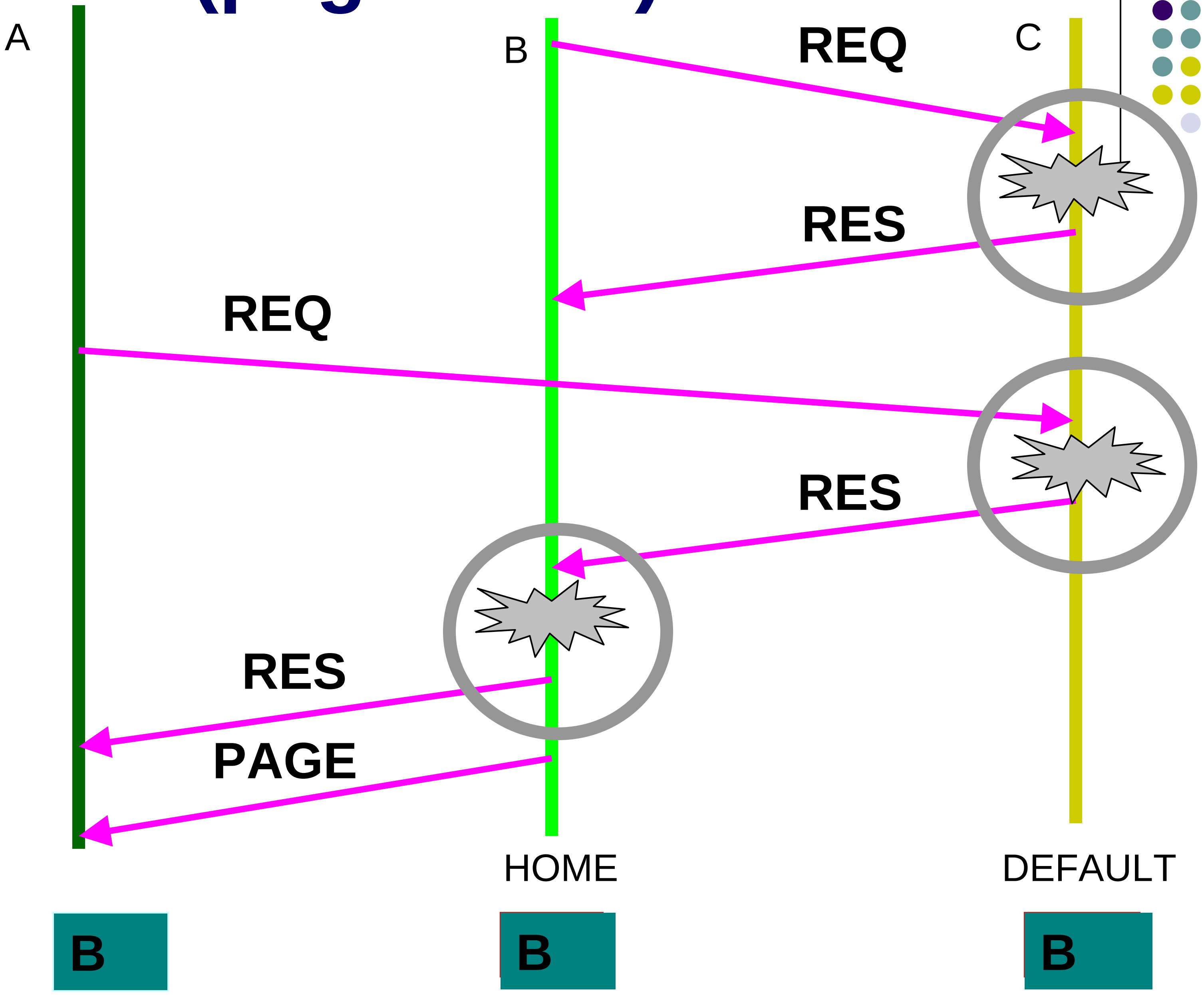
Our Design

- Design consists of 2 protocols
  - ARDMAR (Atomic and RDMA Write)
  - DRAW (Diff using RDMA Write)
- ARDMAR is a synchronous protocol
- DRAW is a hybrid protocol
- NEWGENDSM = ARDMAR + DRAW

NEWGENDSM


HLRC
ASYNC
NEGENDSM
Page
Diff
Lock
Page (ARDMAR)
Diff (DRAW)
Lock

ASYNC (page fetch)



A

B

C

REQ

RES

REQ

RES

RES

PAGE

HOME

DEFAULT

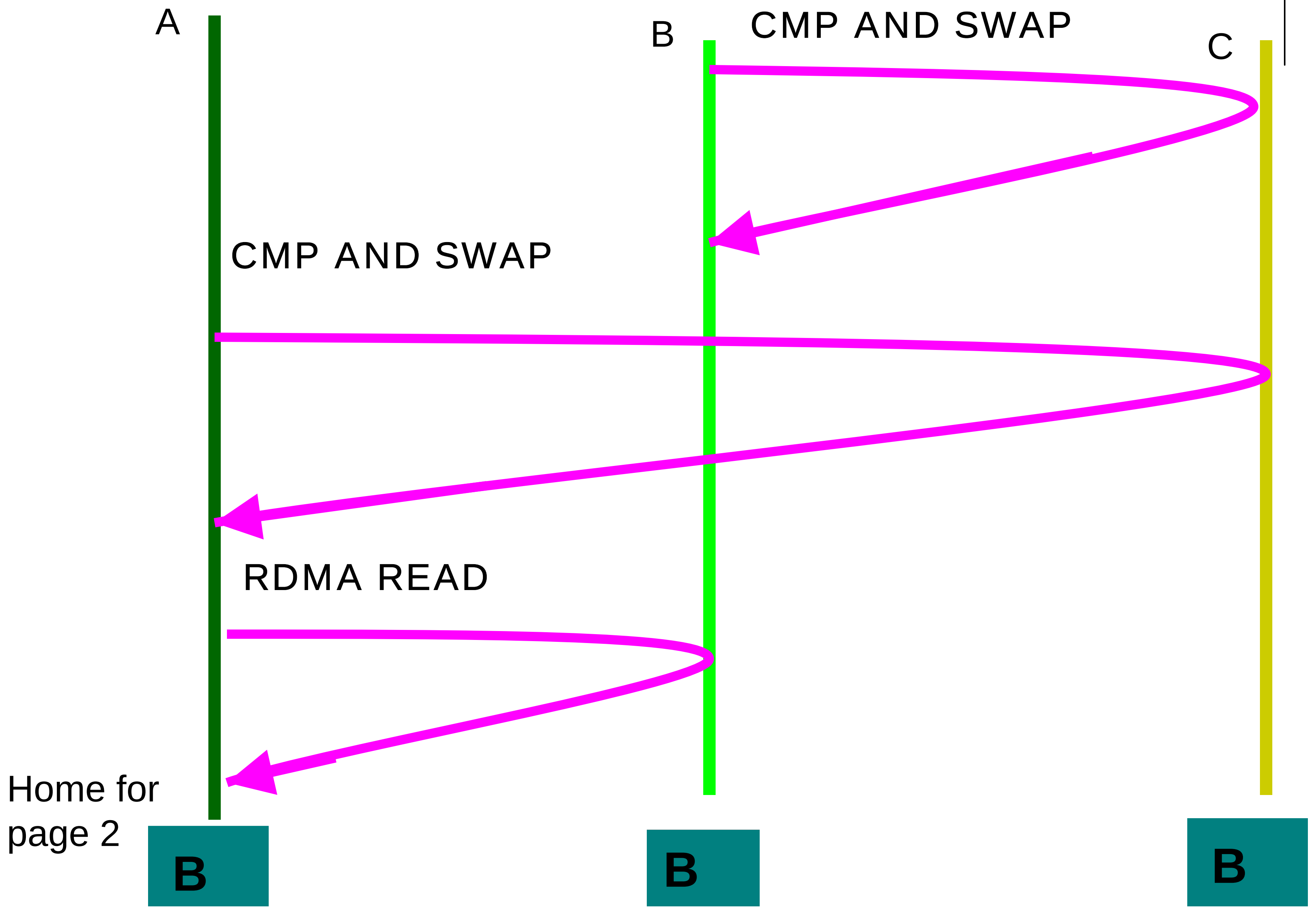
Home for page 2

B

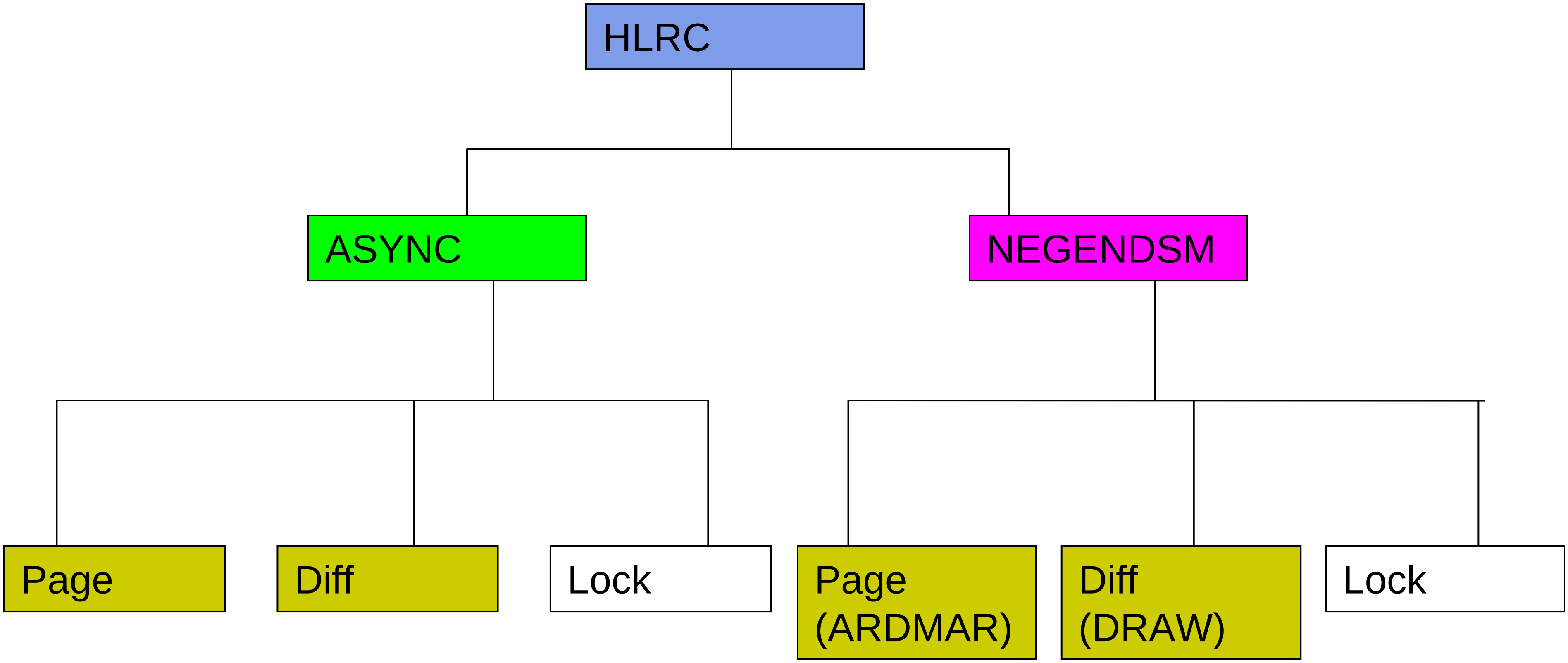
B

B

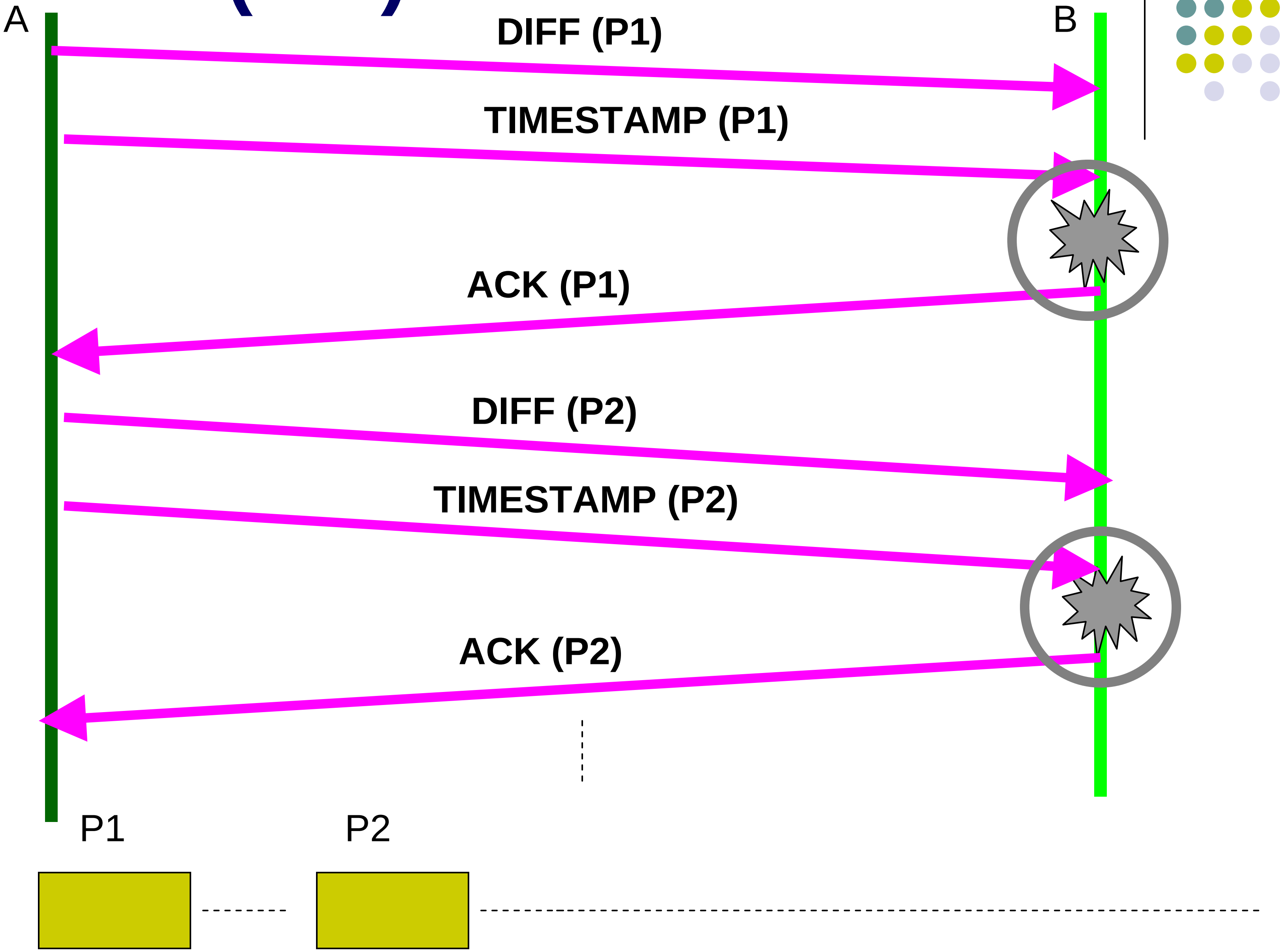
ARDMAR (Atomic and RDMA Write)



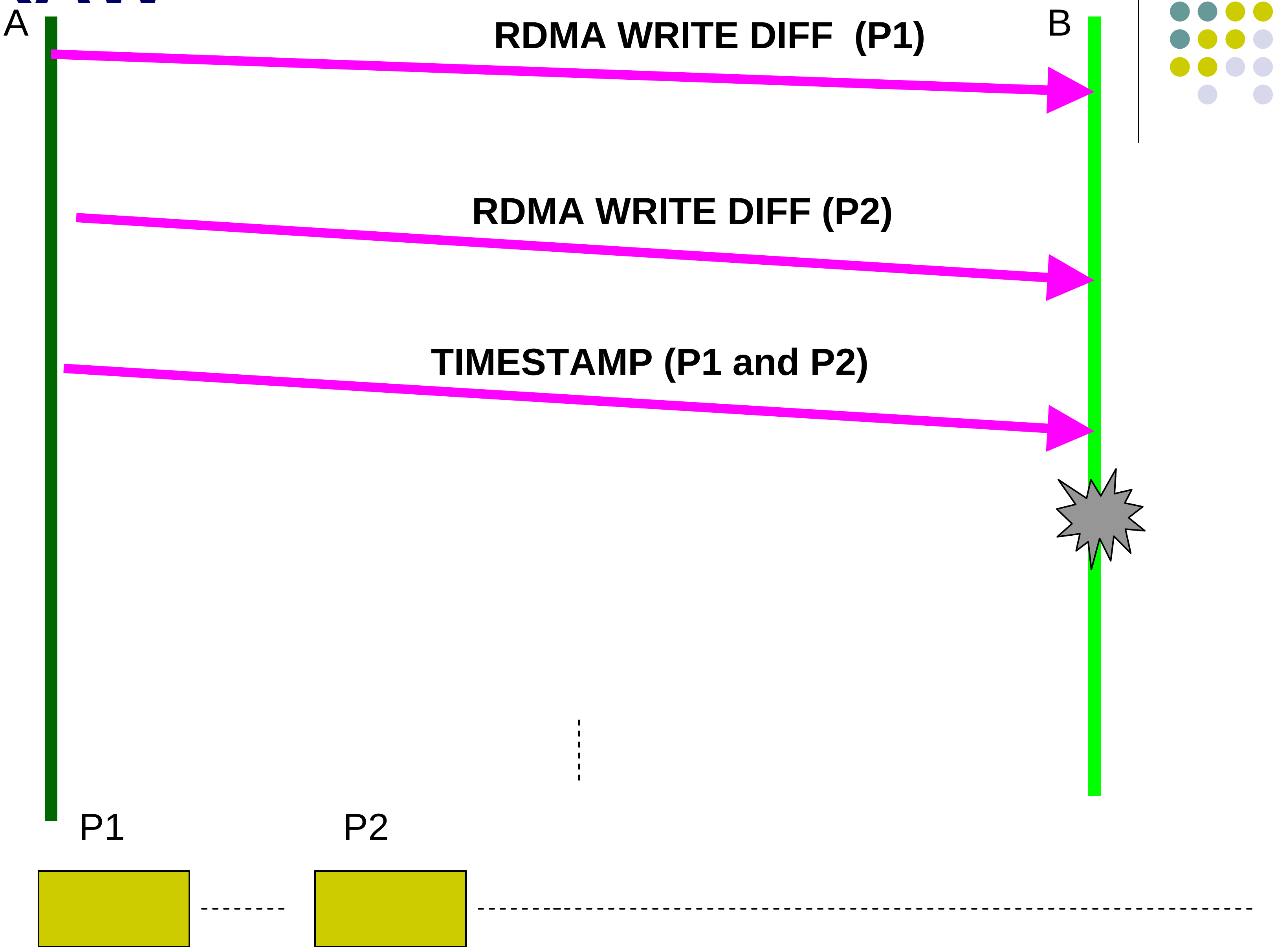
NEWGENDSM


HLRC
ASYNC
NEGENDSM
Page
Diff
Lock
Page (ARDMAR)
Diff (DRAW)
Lock

ASYNC (diff)


A
B
DIFF (P1)
TIMESTAMP (P1)
ACK (P1)
DIFF (P2)
TIMESTAMP (P2)
ACK (P2)
P1
P2

DRAW


A
B
RDMA WRITE DIFF (P1)
RDMA WRITE DIFF (P2)
TIMESTAMP (P1 and P2)
P1
P2

Outline

- Introduction
- Motivation
- Design and Implementation
- Results
- Conclusions
- Future Work

Experimental Setup

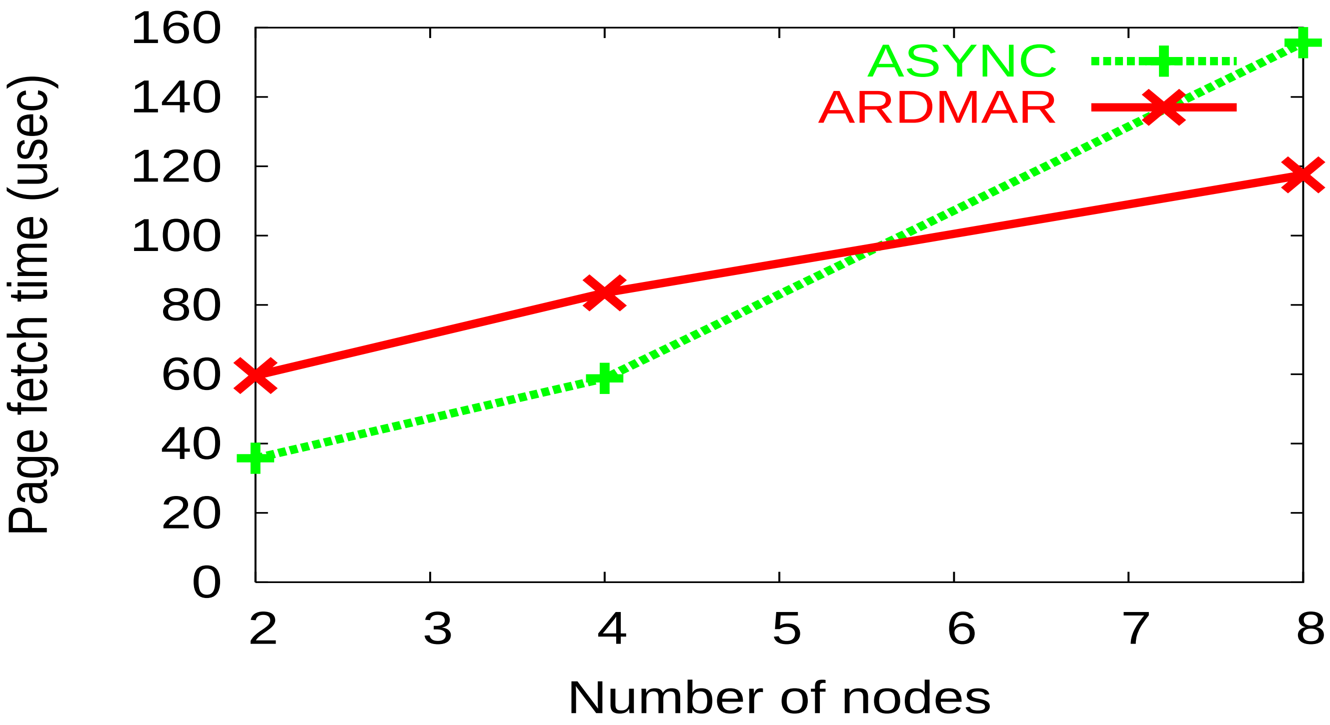
- HLRC/ VIA (Rutgers) modified to work with VAPI
- InfiniScale MT43132 Eight 4X switch
- Mellanox InfiniHost MT23108 DualPort 4X HCA’s
- SuperMicro SUPER P4DL6
  - Dual Pentium Xeon 2.4 GHz
  - 512 MB memory
  - 133 MHz PCI-X bus
- Linux 2.4.7-10 SMP kernel

Evaluation

- Micro-benchmarks (modified from TreadMarks suite)
  - Page →
    - Average time to fetch a page from a home node when a number of nodes are accessing it
  - Diff →
    - Measure Compute Time and Apply Time
    - Small diff (single word) and Large diff (entire page)
- Applications from SPLASH-2 suite (Barnes, TSP, 3Dfft, Radix)

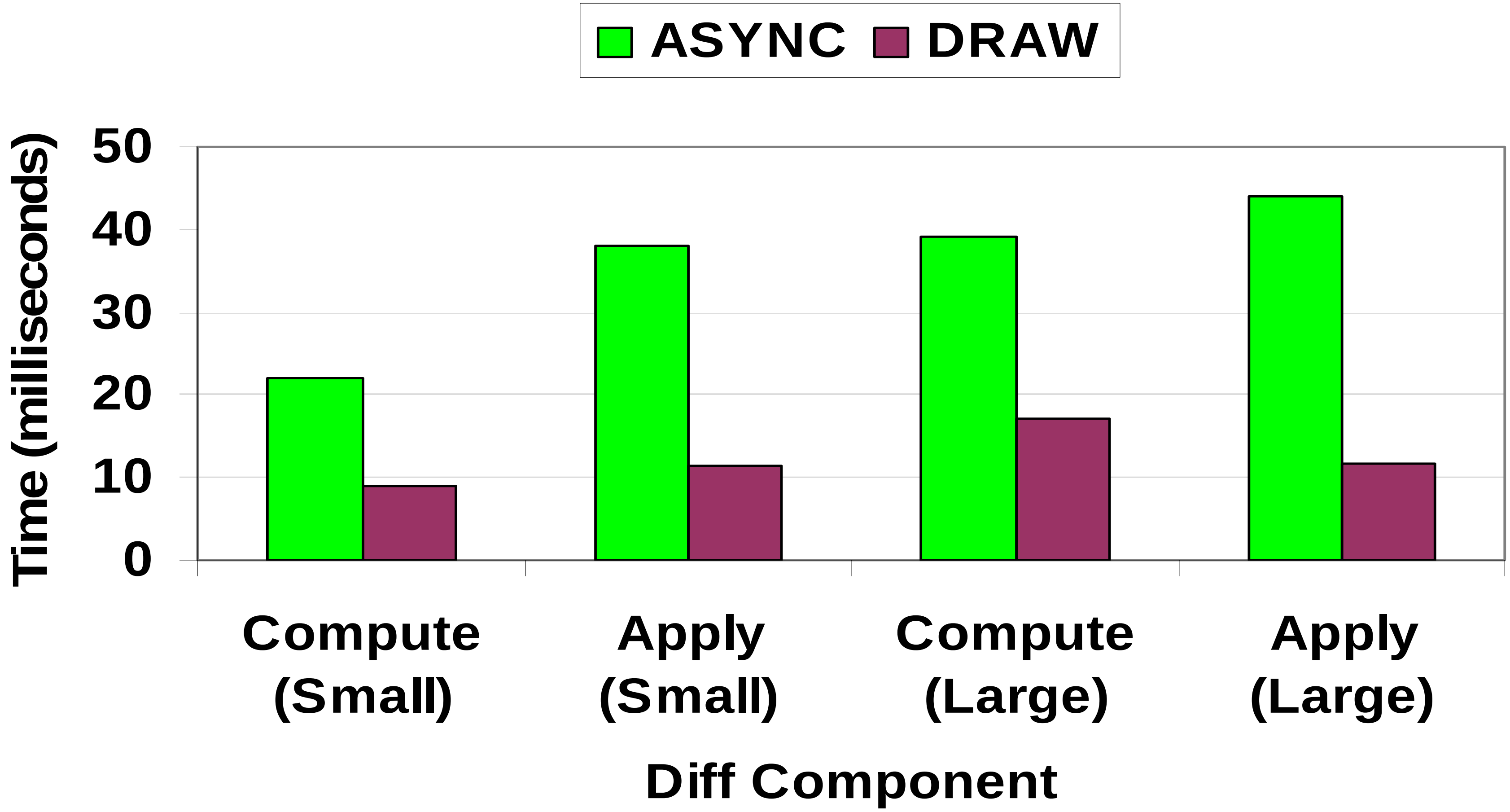
Application	Parameter	Size
Barnes	Bodies	32678
3Dfft	Grid size	128
Radix	Number of keys	2621440
TSP	Tour size	20 (large)

Microbenchmarks (Page)



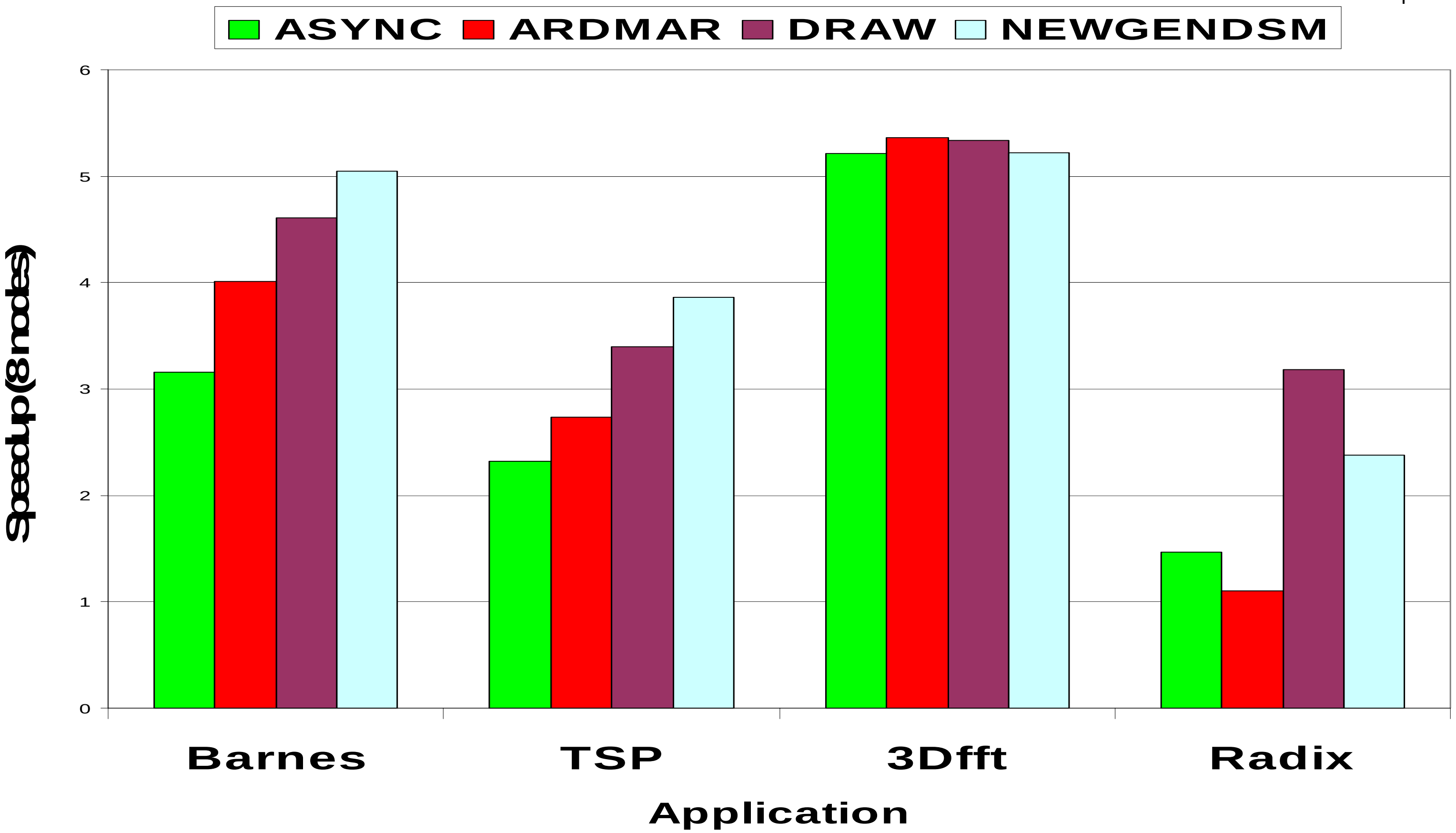
- Page fetching in ARDMAR is lower than ASYNC at 8 nodes

Microbenchmarks (Diff)



- DRAW performs better than ASYNC in all cases

Application Speedup

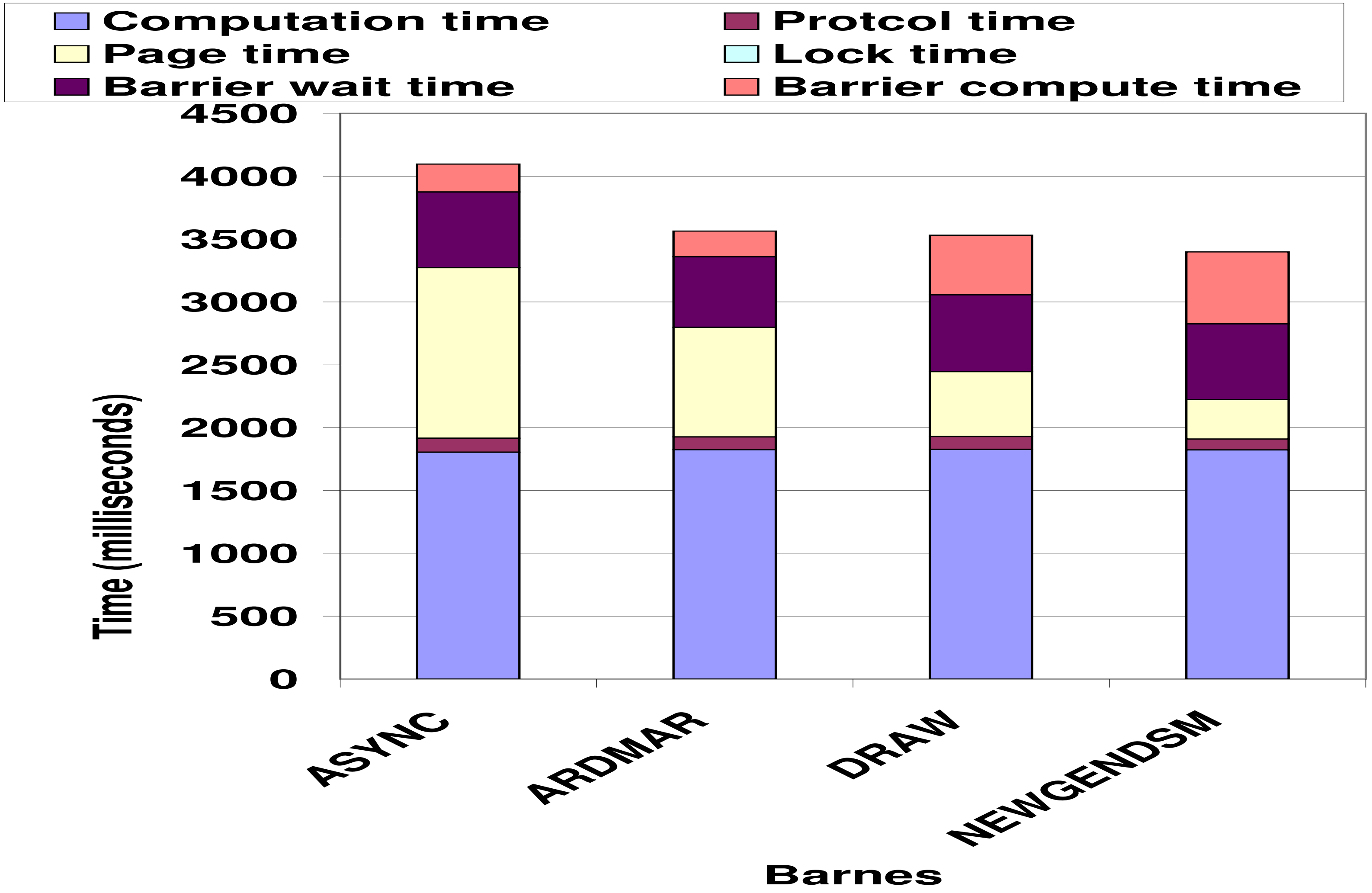


- Speedup w.r.t. sequential running times

•Radix NEWGENDSM speedup 1.63 times ASYNC

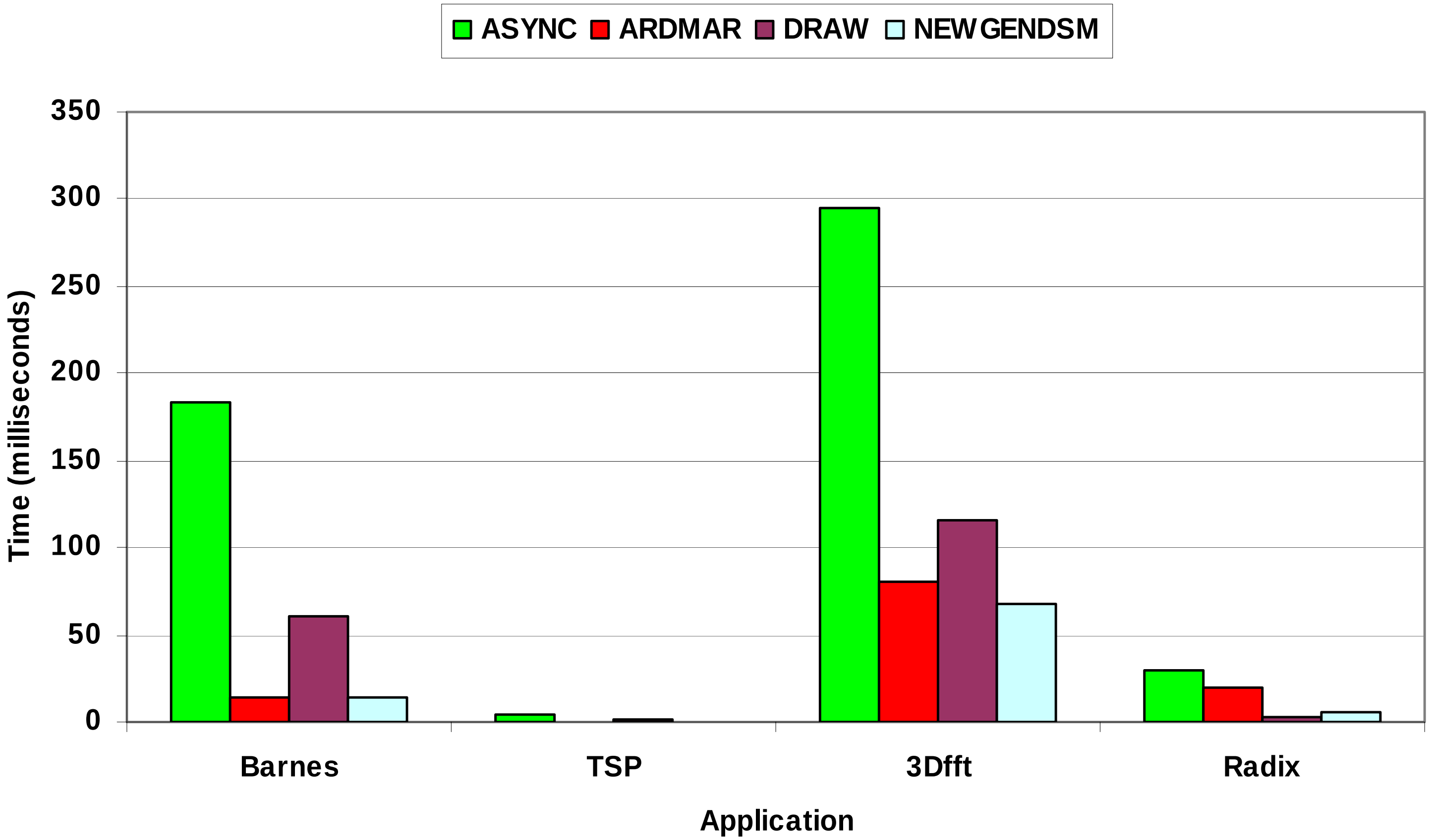
- Barnes NEGENDSM speedup 1.59 times ASYNC

Breakdown



- •Diff time a part of Barrier Compute Time
- •Page time reduced significantly

Asynchronous Handler Time



- Asynchronous handler time substantially reduced for Barnes and 3Dfft

Conclusions

- Explored reducing asynchronous protocol processing time
- Used network features like RDMA Read/Write and atomic operations
- Incorporated in a protocol NEWGENDSM
- Microbenchmark/application level evaluation
- Improvement in parallel speedup upto 1.63

Future Work

- Exploit small message latency to implement “critical word first”
- RDMA Read for “early restart”
- Atomic operations for locking
- Migrating home protocol

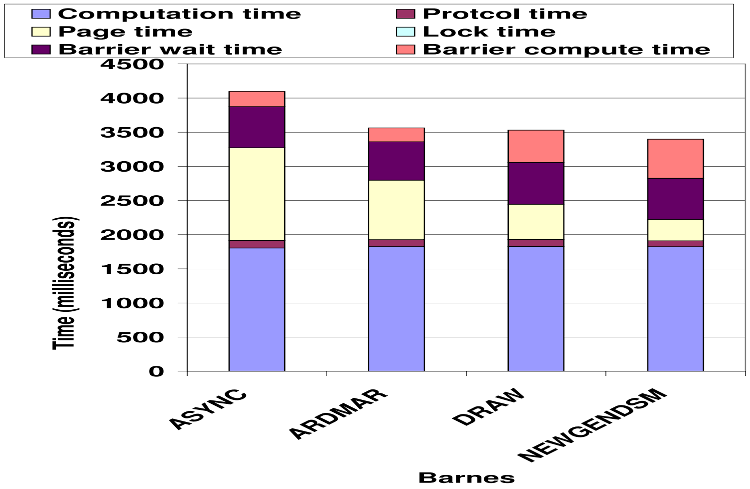
Web Pointers

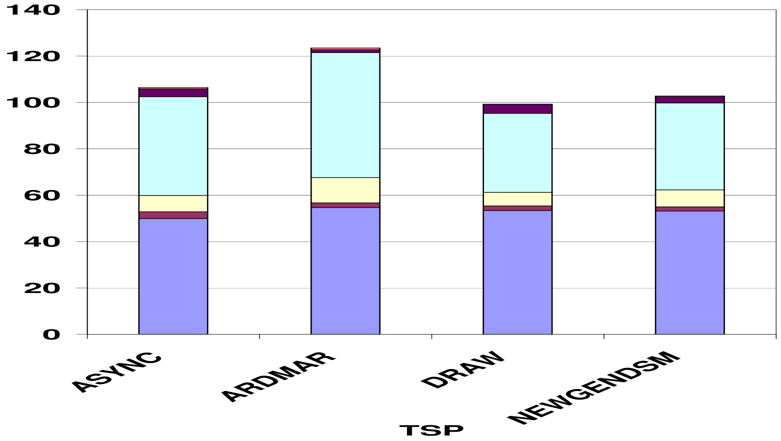
NBC **home page**

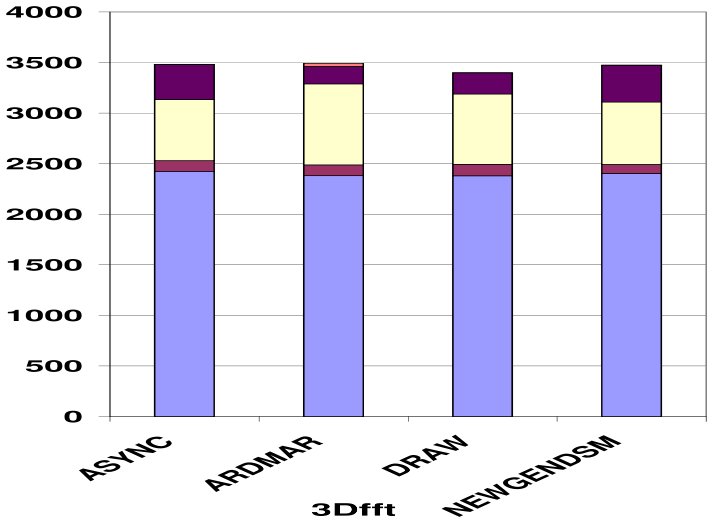
http://nowlab.cis.ohio-state.edu/

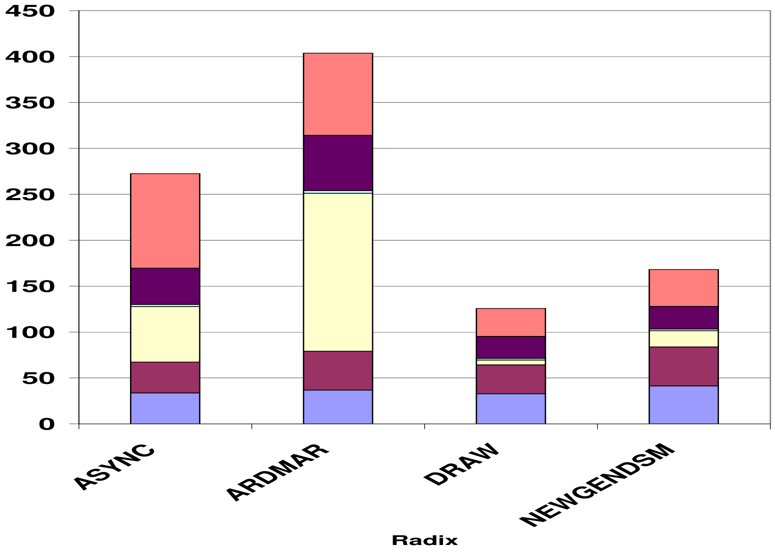
E-mail: {noronha, panda}
@cis.ohio-state.edu

Breakdown









- Page time reduced for Barnes